HoGent

BEDRIJF
EN
ORGANISATIE

Oefening Template pattern

HoGent

Oefening 1

- “Printing”-algoritme voor verschillende types van documenten bestaan uit volgende stappen
  - Header
  - Body
  - Footer

- Werk uit voor een HTML- en een XMLdocument.

- Maak het klassen diagram

Oefening 2 – oplossing 1

- Sorteer Duck-objecten volgens ‘size’. Gebruik een comparator.

```
public class Duck {
   private String name;
   private double weight;
   private double size;
   public Duck(String name, double weight, double size) {
      super();
      this.name = name;
      this.weight = weight;
      this.size = size;   }
   public String toString() {
      return String.format("name= %s, size= %.1f, weight= %.1f", name, size, weight);
   }
   public String getName() {    return name;   }
   public double getSize() {     return size;   }
   public double getWeight() {    return weight;    }
```

Oefening 2 – oplossing 1

```
public class DuckTest {
    @Test
    public void sortBySizeTest() {
        Duck d1 = new Duck("Ducky", 50, 33);
        Duck d2 = new Duck("Greenie", 44, 24);
        Duck d3 = new Duck("Tutsie", 1, 105);
        Duck d4 = new Duck("Lisie", 911, 87);
        Duck[] ducks = {d1, d2, d3, d4};
        Arrays.sort(ducks, Comparator.comparing(Duck::getSize));
        Assert.assertEquals(d2, ducks[0]);
        Assert.assertEquals(d1, ducks[1]);
        Assert.assertEquals(d4, ducks[2]);
        Assert.assertEquals(d3, ducks[3]);
    }
}
```

Op welk design pattern is dit gebaseerd?

Oefening 2 – oplossing 2

```
public class DuckTest {
   @Test
   public void sortBySizeTest() {
      Duck d1 = new Duck("Ducky", 50, 33);
      Duck d2 = new Duck("Greenie", 44, 24);
      Duck d3 = new Duck("Tutsie", 1, 105);
      Duck d4 = new Duck("Lisie", 911, 87);
      Duck[] ducks = {d1, d2, d3, d4};
      Arrays.sort(ducks);
      Assert.assertEquals(d2, ducks[0]);
      Assert.assertEquals(d1, ducks[1]);
      Assert.assertEquals(d4, ducks[2]);
      Assert.assertEquals(d3, ducks[3]);
   }
}
```

Op welk design pattern is dit gebaseerd?

Template methode in het echt

- Sorteren met de Template methode
  - De ontwerpers van de Java klasse Array hebben een handige template methode voor het sorteren voorzien.
    - Onderstaande code is voor de eenvoud wat ingekort
    - Het zijn 2 methoden, samen leveren ze de sorteerfunctionaliteit
    - Sort() is een hulpmethode.
      - Maakt kopie van array en geeft deze door aan de mergeSort
      - Geeft ook de lengte van de array door aan mergeSort
      - Zorgt ervoor dat de mergeSort bij het eerste element begint

```
public static void sort(Object[] a) {
    Object aux[] = (Object[])a.clone();
    mergeSort(aux, a, 0, a.length, 0);
}
```

Template methode in het echt

- Sorteren met de Template methode
  - klasse Array met template methode voor het sorteren
    - mergeSort() bevat het sorteeralgoritme en rekent op de implementatie van de methode compareTo() om het algoritme te voltooien.
      - Beschouw deze code als de templatemethode
      - swap is een concrete methode die in de klasse Array gedefinieerd is
      - mergeSort() (en dus sort()) rekent op een Comparable klasse om te zorgen voor een implementatie van compareTo(). Die methode moeten we implementeren

```
private static void mergeSort(Object src[], Object dest[],
             int low, int high, int off)
{

    for (int i=low; i<high; i++){
        for (int j=i; j>low &&
             ((Comparable)dest[j-1]).compareTo((Comparable)dest[j])>0; j--)
        {
            swap(dest, j, j-1);
        }
    }
    return;
}
```

Eerste eend

Eend waarmee vergeleken wordt

Om een array te sorteren moet je 2 items 1 voor 1 vergelijken tot de gehele lijst gesorteerd is

    - Voor meer detail : zie broncode Sun